To read the article online, visit http://www.4GuysFromRolla.com/articles/062508-1.aspx

Examining ASP.NET's Membership, Roles, and Profile - Part 11

By Scott Mitchell

Introduction

Many websites that support user account allow anyone to create a new account, but require new users to undergo some form of verification before their account is activated. A common approach is to send an email to the newly created user with a link that, when visited, activates their account. This approach ensures that the email address entered by the user is valid (since it is sent to that user's email address). This workflow not only ensures the valid data entry, but also helps deter automated spam bots and abusive users.

In past installments of this article series we've seen how to use the CreateUserWizard control to allow users to create new accounts. By default, the user accounts created by the CreateUserWizard control are activated; new users can login immediately and start interacting with the site. This default behavior can be customized, however, so that new accounts are disabled. A disabled user cannot log into the site; therefore, there needs to be some manner by which a newly created user can have her account enabled.

There are many ways by which an account may be activated. You could have each account manually verified by an administrative user. If your site requires users to pay some sort of monthly fee or annual due, you could have the account approved once the payment was successfully processed. As aforementioned, one very common approach is to require the user to visit a link sent to the email address they entered when logging on. This article explores this latter technique. Read on to learn more!

Disabling Newly Created User Accounts

In order to require some form of activation before a newly created account is activated we need to make sure that newly created accounts are disabled. By default, new accounts created with the CreateUserWizard control are activated, meaning that the just-created user can log into the site and use the members-only functionality. The good news is that configuring the CreateUserWizard control so that new accounts are inactive is a cinch - simply set the control's `DisableCreatedUser` property to True.

When disabling new user accounts it's also a good idea to let the user know that they cannot login until their account is activated, and to explain what steps are required to activate their account. This information can best be conveyed in the CreateUserWizard's Complete Wizard Step, which is the screen that appears after the user enters their username, email address, password, and other information and clicks the "Create User" button. The Complete Wizard Step, by default, displays the text: "Your account has been successfully created." You can modify this display via the CreateUserWizard control's `CompleteSuccessText` property.

The download available at the end of this article includes a complete working demo. I've customized the CreateUserWizard control in `CreateAccount.aspx` to set the `DisableCreatedUser` and `CompleteSuccessText` properties to True and "Your account has been created, but before you can login you must first verify your email address. A message has been sent to the email address you specified. Please check your email inbox and follow the instructions in that email to verify your account", respectively.

Sending an Email with a Verification URL to the Newly Created User Account

In addition to disabling the newly created user account, we also need to send them an email that contains a URL that, when clicked, activates their account. Before we focus on what, precisely, must be in this email, let's first look at how to send an email to a newly created user account. The CreateUserWizard control has a `MailDefinition` property that you can set that specifies information about an email to send upon successfully creating a new user account. The `MailDefinition` property has subproperties like `From`, `Subject`, and `BodyFileName`. The mail settings specified in `Web.config` are used to physically send the email; see Sending Email in ASP.NET 2.0 for more information on this necessary configuration.

The `BodyFileName` specifies a file that contains the body of the email to send. This file can include placeholders to dynamically inject values when the email is sent. The CreateUserWizard automatically will replace the placeholders `<%UserName%>` and `<%Password%>` with the newly created user's username and password. Just before the email is sent, the CreateUserWizard control raises its `SendingMail` event. You can create an event handler for this event to programmatically replace any custom placeholders with appropriate values.

The download at the end of this demo includes an email body template file in the `~/EmailTemplates` folder named `NewAccountTemplate.htm`. This template file specifies an HTML-formatted email body. It's pertinent sections follow:

```
<p>
   Hello, <%UserName%>. You are receiving this email because you recently created a new
account at my
   site. Before you can login, however, you need to first visit the following link:
</p>
<p>
   <a href="<%VerifyUrl%>"><%VerifyUrl%></a>
</p>
```

Note the `<%VerifyUrl%>` placeholder. This is an example of a custom placeholder. We need to replace it with the URL the user will visit to verify their email address and activate their account. In the demo at the end of this article I created a page named `Verify.aspx`. This page is passed the user's `UserId` value (a GUID) through the querystring, like: `Verify.aspx?ID=a4013b37-389f-4fa6-ad02-c4144cffa511`. It then looks up that user information and approves them.

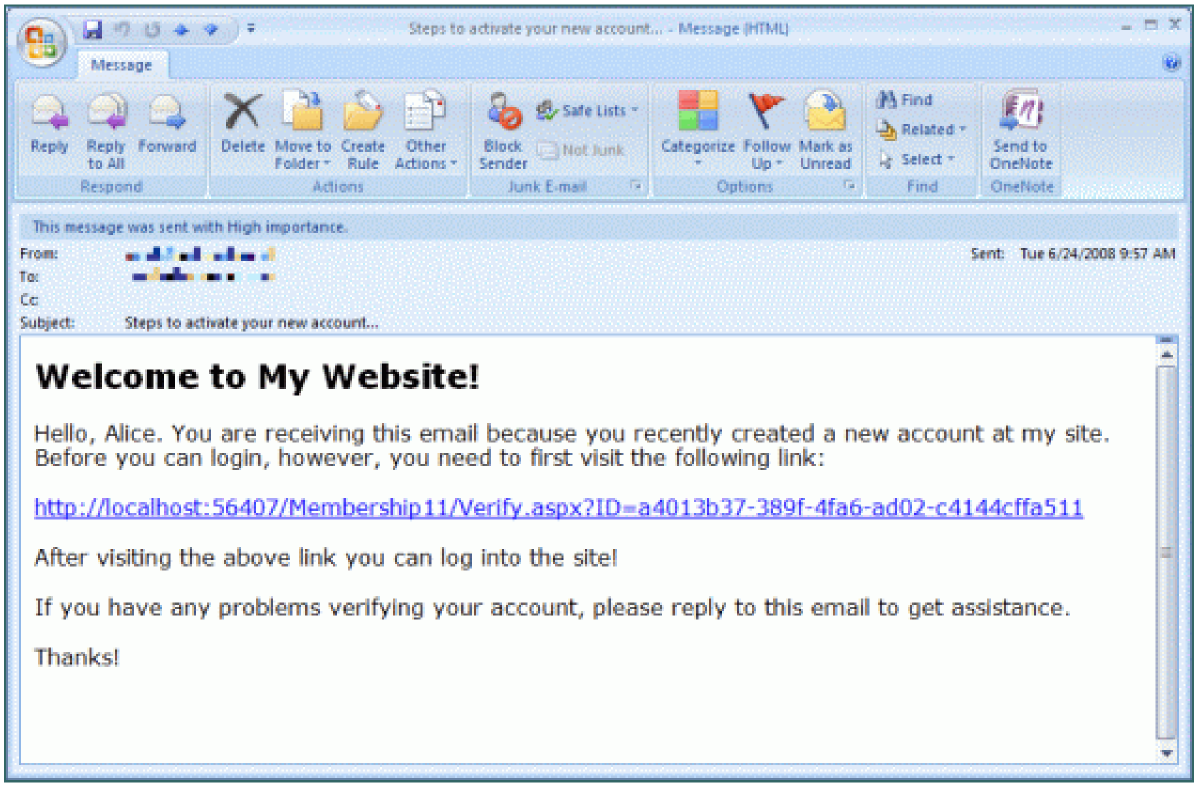
Consequently, we need to replace the `<%VerifyUrl%>` placeholder with `Verify.aspx?ID=`*`UserId`*. To accomplish this, create an event handler for the CreateUserWizard control's `SendingMail` event and add the following code:

```
Protected Sub CreateUserWizard1_SendingMail(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.MailMessageEventArgs) Handles CreateUserWizard1.SendingMail
   Dim userInfo As MembershipUser = Membership.GetUser(CreateUserWizard1.UserName)

   'Construct the verification URL
   Dim verifyUrl As String = Request.Url.GetLeftPart(UriPartial.Authority) &
Page.ResolveUrl("~/Verify.aspx?ID=" & userInfo.ProviderUserKey.ToString())

   'Replace <%VerifyUrl%> placeholder with verifyUrl value
   e.Message.Body = e.Message.Body.Replace("<%VerifyUrl%>", verifyUrl)
End Sub
```

This code starts by getting information about the just created user via `Membership.GetUser(`*`username`*`)`. (The just created user's username is available from the CreateUserWizard control's `UserName` property.)

Next, the verification URL is constructed, using the just created user's `UserId` value as the value of the `ID` querystring field. Next, the email message body - accessible through `e.Message.Body` - is assigned its value with the placeholder `<%VerifyUrl%>` replaced with the verification URL value.

After creating a new account, the user will receive an email in their inbox with a URL to the verification page.



This message was sent with High importance.
From: [redacted]
To: [redacted]
Cc:
Subject: Steps to activate your new account...
Sent: Tue 6/24/2008 9:57 AM

Welcome to My Website!

Hello, Alice. You are receiving this email because you recently created a new account at my site. Before you can login, however, you need to first visit the following link:

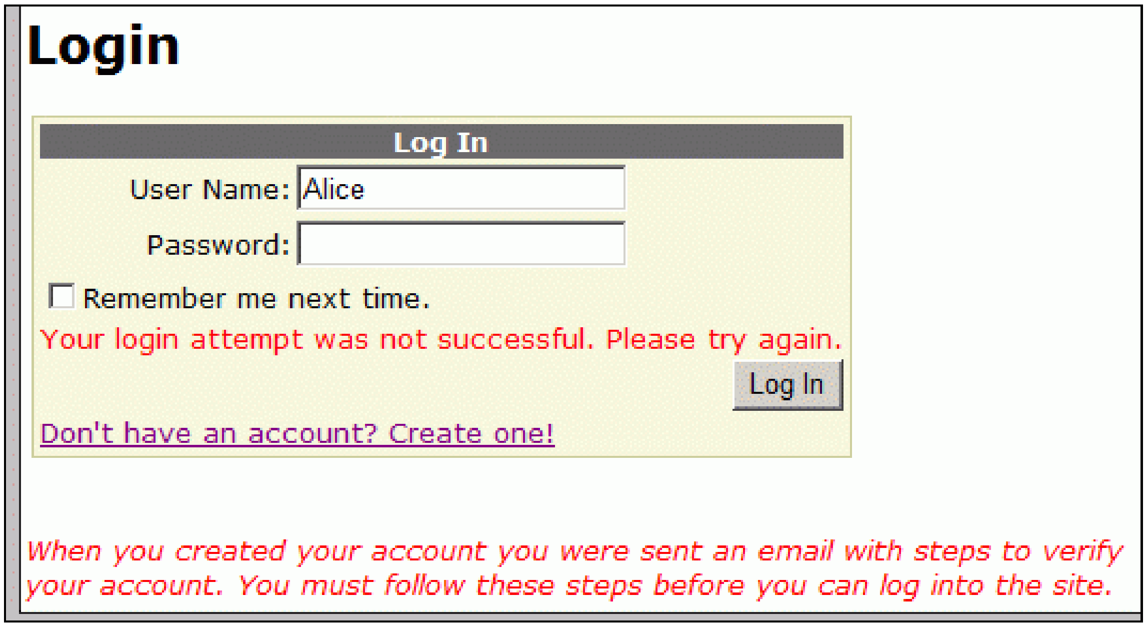
http://localhost:56407/Membership11/Verify.aspx?ID=a4013b37-389f-4fa6-ad02-c4144cffa511

After visiting the above link you can log into the site!

If you have any problems verifying your account, please reply to this email to get assistance.

Thanks!

Keep in mind that after creating an account, but before verifying it, a user cannot log into the site because their account is disabled. If they attempt to login, they'll get a message saying that their credentials are invalid. In Part 4 of this article series we saw how to enhance the Login control to display a more meaningful message when a user was unable to login. I've used these techniques to enhance the login page (`Login.aspx`) to display a special message if the user cannot login because their account has not yet been activated.



User Name: Alice
Password:

Your login attempt was not successful. Please try again.

Don't have an account? Create one!

When you created your account you were sent an email with steps to verify your account. You must follow these steps before you can log into the site.

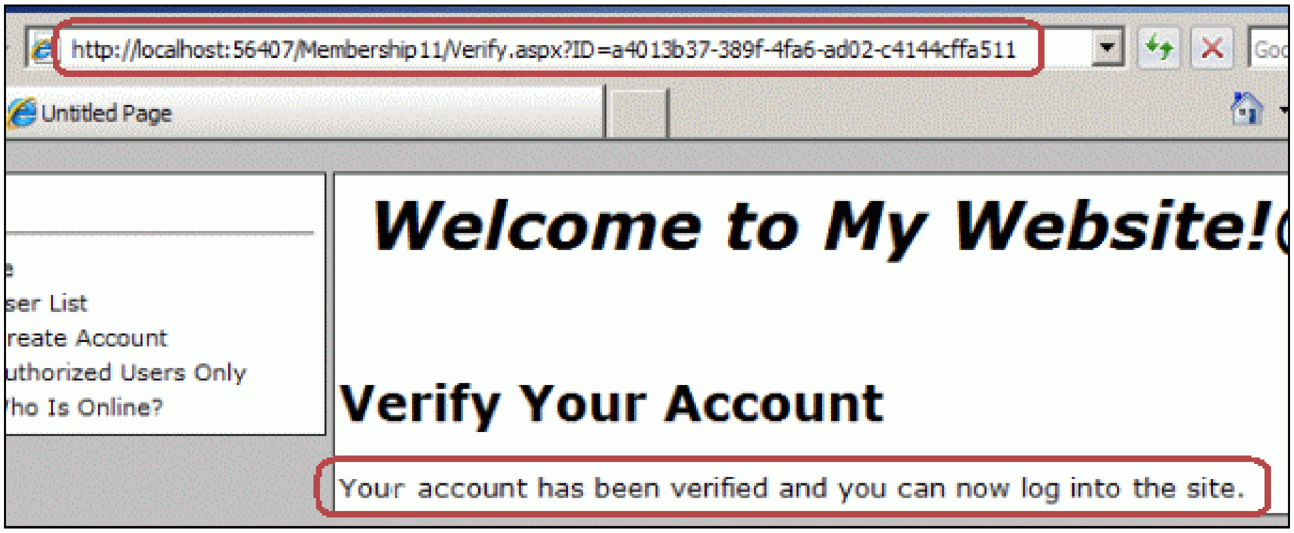
Creating the Verification Page

At this point new user accounts are automatically disabled and are sent an email with a link to the verification page, `Verify.aspx`. All that remains is to create the `Verify.aspx` page. When visited, this page must activate the user account. This is accomplished via the following code, which is located in the `Page_Load` event handler of `Verify.aspx`:

```
'Make sure that a valid querystring value was passed through
If String.IsNullOrEmpty(Request.QueryString("ID")) OrElse Not
Regex.IsMatch(Request.QueryString("ID"), "[0-9a-f]{8}\-([0-9a-f]{4}\-){3}[0-9a-f]{12}")
Then
   InformationLabel.Text = "An invalid ID value was passed in through the querystring."
Else
   'ID exists and is kosher, see if this user is already approved
   'Get the ID sent in the querystring
   Dim userId As Guid = New Guid(Request.QueryString("ID"))

   'Get information about the user
   Dim userInfo As MembershipUser = Membership.GetUser(userId)
   If userInfo Is Nothing Then
      'Could not find user!
      InformationLabel.Text = "The user account could not be found in the membership
database."
   Else
      'User is valid, approve them
      userInfo.IsApproved = True
      Membership.UpdateUser(userInfo)

      'Display a message
      InformationLabel.Text = "Your account has been verified and you can now log into
the site."
   End If
End If
```

The above code starts by ensuring that an `ID` value was passed through the querystring, and that it matches the pattern expected by a GUID. Next, it uses the `Membership.GetUser(`*`userId`*`)` method to get information about the user corresponding to the `UserId` passed through the querystring. If no user account is returned, a message is displayed. If a user account is found with that `UserId` value, however, it is activated. To activate a user account set the `IsApproved` property to True and then update the user.

That's all there is to it! After visiting this page, a user is activated and can now log into the site. The screen shot below shows the `Verify.aspx` page when visited through a browser. Note that the `UserId` is passed along in the querystring; after activating the account the message "Your account has been verified and you can now log into the site" is displayed.


http://localhost:56407/Membership11/Verify.aspx?ID=a4013b37-389f-4fa6-ad02-c4144cffa511

Untitled Page

Welcome to My Website!

ser List
reate Account
uthorized Users Only
'ho Is Online?

Verify Your Account

Your account has been verified and you can now log into the site.

Security Considerations

Because the `Verify.aspx` page automatically approves a user account based on the `UserId` value passed through the querystring, you may worry that an unscrupulous user may be able to circumvent email verification by visiting `Verify.aspx` and "guessing" an appropriate `UserId` value. However, the chances of this are very, very, very remote. The `UserId` is a globally unique identifier (GUID) value, which is a 128-bit value. This means that there are 2^{128} possible GUID values, which translates into roughly 340,000,000,000,000,000,000,000,000,000,000,000,000 possible values. That's a lot! The chances of a user "stumbling" upon a GUID value are minuscule. They'd have a better chance getting struck by lightning, giving birth to quadruplets, and winning the lottery all in the same day.

One possible enhancement of the `Verify.aspx` implementation we examined would be to add some sort of time-based expiry. For example, you could have `Verify.aspx` only approve a user visiting if the user's account creation date (available via the `MembershipUser` object's `CreationDate` property) is less than one week old. This would be useful if you may later deactivate abusive or spammy users, so that they cannot reactivate themselves by revisiting `Verify.aspx`.

Conclusion

The CreateUserWizard control creates new user accounts as approved (by default), but can be configured to disable new accounts. This approach is useful if there is some step (or steps) that must be complete before a new user can log into the site. This may involve an administrator approving a user or, as we saw in this article, through some user actions, such as verifying their email address.

Once the necessary steps have been completed, to activate a user's account get a `MembershipUser` object for that user account, set its `IsApproved` property to True, and then save the updated user account via a call to `Membership.UpdateUser`. After being marked approved, the user can login.

Happy Programming!

- By Scott Mitchell

Further Reading

- Forms Authentication, Authorization, Membership, and Roles Tutorials (includes VB & C# versions!)
- Sending Email in ASP.NET 2.0
- Improving the Login Experience
- Unlocking and Approving User Accounts (VB version) (C# version)

Attachments

- Download the code used in this article

Article Information	
Article Title:	ASP.NET.Examining ASP.NET's Membership, Roles, and Profile - Part 11
Article Author:	Scott Mitchell
Published Date:	June 25, 2008
Article URL:	http://www.4GuysFromRolla.com/articles/062508-1.aspx

---